

# Customer Requirements Specification

(Lastenheft)

(TINF18C, SWE I Praxisprojekt 2019/2020)

Project

AMLEngine-DLL Interfaces

Customer:

Rentschler & Ewertz  
Rotebühlplatz 41  
70178 Stuttgart

Supplier:

Team 4 Joshua, Kevin, Krister, Lucas, Markus, Robin  
Rotebühlplatz 41  
70178 Stuttgart

Version	Date	Author	Comment
0.1	11.10.2019	Markus	created
0.2	14.10.2019	Markus	continued
0.3	17.10.2019	Markus	Business Processes and Use Cases
0.4	18.10.2019	Markus	Functional requirements
0.5	24.10.2019	Markus	Changes to non-functional requirements
0.6	25.10.2019	Markus	Formulated whole sentences
0.7	29.10.2019	Markus	Stylistic improvements
0.8	31.10.2019	Markus	Deleted open points
1.0	01.10.2019	Markus	Final check and last minor changes before publication
2.0	05.05.2020	Markus	Final version of this document with some minor changes before the final submission.

# Contents

Goal	3
Product Environment	3
Product Usage	4
Business Processes	4
<BP.001>: C++ Wrapper	4
<BP.002>: JavaScript Wrapper	5
<BP.003>: Console Application	6
Use Cases	7
<UC.001> C++ Wrapper	7
<UC.002> JavaScript Wrapper	8
<UC.003> Validation using Console Application	9
<UC.004> (Un-)packing using Console Application	9
Features	10
/LF10/C++ Functions	10
/LF20/C++ usability	10
/LF30/JavaScript Functions	10
/LF40/JavaScript usability	10
/LF50/Import	10
/LF60/(De-)Compression	10
/LF70/Validation	10
Product Data	11
/LD10/AML file	11
/LD20/AMLX file	11
Other Product Characteristics	11
/NF10/Documentation	11
/NF20/Installation	11
/NF30/Files	11
/NF40/System Environment	11
References	12

# Goal

The goal of this project is to develop a software which allows easy usage of the AMLEngine<sup>[1]</sup>, more specifically the AML.Engine.dll. AutomationML is an open standard based on XML<sup>[2]</sup>. It allows easier modelling of automation processes by reducing the number of redundant files which are normally created along every step of the design process. This standard is developed by Automation e.V. A DLL is a file that contains precompiled code which can be easily linked into different projects similar to a library.

This project can be split into three parts. The first part is the usage of the AML.Engine.dll inside of a C++ project. To achieve this, certain project settings have to be adjusted as well as special compiler settings. Those setup steps have to be precisely specified inside of a document to support developers who want to create such a project.

The second part of this project is the usage of the AML.Engine.dll inside of JavaScript code. Similar to the usage of the DLL inside of a C++ project, this will also require a certain project setup as well as some custom code to call the functions contained inside of the DLL.

The third part of the project will be the development of a console application which can be used to pack or unpack AMLX files, as well as to verify those files.

This software is intended to be used by developers who want to integrate AMLEngine functions into their code.

# Product Environment

AutomationML is used to design automation processes such as assembly lines. It is meant to be used along the whole life cycle of a product, thus aiming to help during the design process but also during installation and maintenance. An AML file is based on a CAEX<sup>[3]</sup> file which uses XML notation and is used to specify different classes and components of an automated plant in a given hierarchy. It can include references to external files containing further specifications of specialized parts. The AML.Engine.dll can be used to work with and modify AML files.

The current situation is that AML files can be edited by directly working on the file itself. This requires opening the file in a text editor and manually writing all the XML tags and properties. This can be a difficult and complex process that will likely lead to many errors. Another possibility to edit AML files is to use the AutomationML Editor. This

editor allows easier manipulation of the file and provides a graphical user interface. However, this editor is still under development and many bugs make working with it challenging.

The software developed for this project is meant to make the usage of AML files easier, by paving the way for the development of additional editing tools that can be used to work on AML files. This can be done by providing new and improved ways to access the functions contained inside of the AML.Engine.dll.

## Product Usage

The following business processes, use cases and features shall be supported by the system.

### Business Processes

#### <BP.001>: C++ Wrapper

A developer wants to use the AML.Engine.dll in their C++ project. To do so they need a wiki containing all important setup steps and project settings.

Triggering Event:	Developer wants to use the AML.Engine.dll in a C++ project
Result:	Developer can use the AML.Engine.dll in their project
Involved Roles:	Developer, C++ project, wiki/documentation, AML.Engine.dll

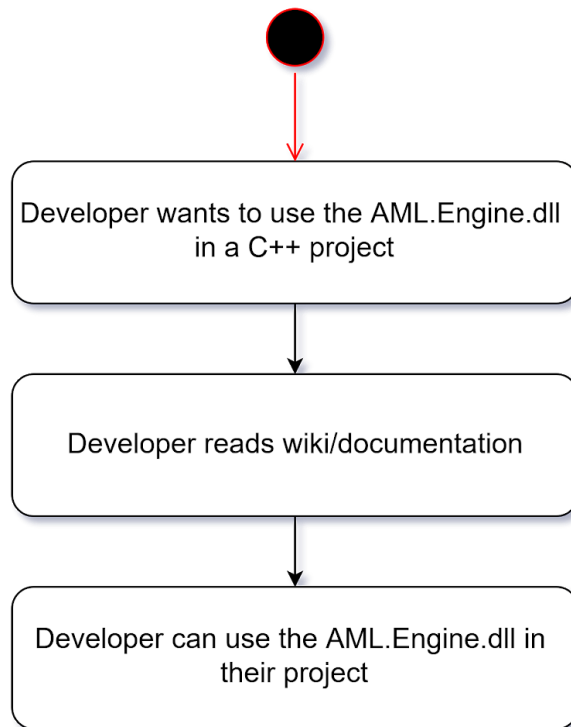


Figure 1: activity diagram for BP.001

### <BP.002>: JavaScript Wrapper

A developer wants to use the AML.Engine.dll in their Node project. To do so, they need a way to access the functions contained inside of the AML.Engine.dll.

Triggering Event:	Developer wants to use AML.Engine.dll inside of a Node project
Result:	Developer can use the AML.Engine.dll inside of their Node project
Involved Roles:	Developer, wrapper, wiki/documentation, AML.Engine.dll, Node project

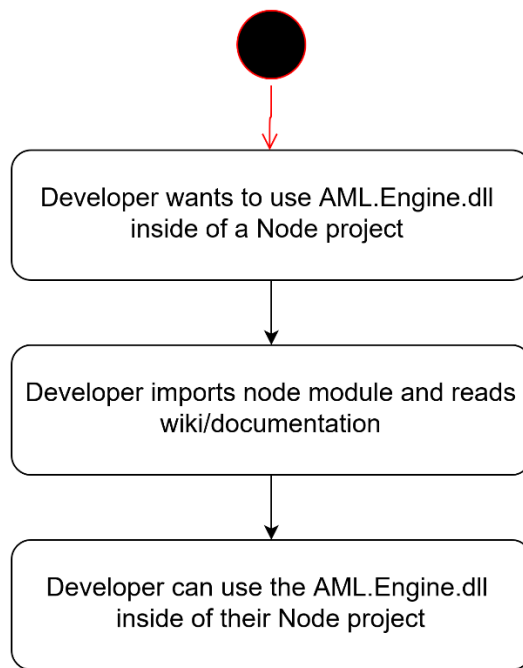


Figure 2: activity diagram for BP.002

### <BP.003>: Console Application

A developer or a system architect wants to check whether their AML file is valid. This might also require unpacking an AMLX file first and repack it afterwards.

Triggering Event:	User wants to validate, pack or unpack an AMLX or AML file
Result:	<p>User gets confirmation that their file is valid or invalid.</p> <p>User gets the content of the unpacked AMLX file or gets the new AMLX file that contains all the files they wanted to combined into a single file.</p>
Involved Roles:	console application, user, AML.Engine.dll

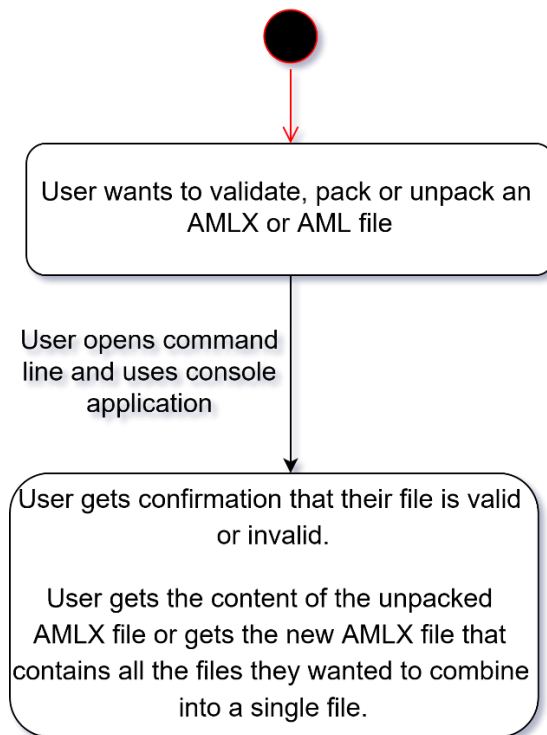


Figure 3: activity diagram for BP.003

## Use Cases

### <UC.001> C++ Wrapper

<b>Related Business Process:</b>	<BP.001>: C++ Wrapper
<b>Use Cases Objective:</b>	Developer implements a C++ program that uses the AML.Engine.dll
<b>System Boundary:</b>	Program wide, including imported libraries
<b>Precondition:</b>	The correct version of the AML.Engine.dll has to be installed.
<b>Postcondition on success:</b>	The developer was able to develop a functioning program which uses functions contained inside of the AML.Engine.dll



<b>Involved Users:</b>	AML.Engine.dll, C++ project, developer
<b>Triggering Event:</b>	Developer wants to use functions contained inside of the AML.Engine.dll inside of their C++ project.

### <UC.002> JavaScript Wrapper

<b>Related Business Process:</b>	<BP.002>: JavaScript Wrapper
<b>Use Cases Objective:</b>	Developer can implement a JavaScript project that can access functions contained inside the AML.Engine.dll.
<b>System Boundary:</b>	Program wide, including imported modules
<b>Precondition:</b>	The correct version of the AML.Engine.dll needs to be installed
<b>Postcondition on success:</b>	Developer was able to develop a project that can access the functions contained inside the AML.Engine.dll.
<b>Involved Users:</b>	AML.Engine.dll, wrapper, JavaScript project, developer
<b>Triggering Event:</b>	Developer wants to use the AML.Engine.dll in their project

### <UC.003> Validation using Console Application

<b>Related Business Process:</b>	<BP.003>: Console Application
<b>Use Cases Objective:</b>	The console is application is used to validate an AML file
<b>System Boundary:</b>	Console
<b>Precondition:</b>	All required files need to be downloaded to the user's computer.
<b>Postcondition on success:</b>	In case of an invalid file the error will be printed out. A valid file will return a success message.
<b>Involved Users:</b>	AML file, console application, User, AML.Engine.dll
<b>Triggering Event:</b>	User wants to validate their AML file

### <UC.004> (Un-)packing using Console Application

<b>Related Business Process:</b>	<BP.003>: Console Application
<b>Use Cases Objective:</b>	The console application can be used to pack and unpack AML and AMLX files respectively.
<b>System Boundary:</b>	Console
<b>Precondition:</b>	All required files need to be downloaded to the user's computer.

<b>Postcondition on success:</b>	The file was successfully packed or unpacked
<b>Involved Users:</b>	AML/AMLX file, console application, user, AML.Engine.dll
<b>Triggering Event:</b>	User wants to pack or unpack AML or AMLX files

## Features

### **/LF10/C++ Functions**

The C++ wrapper shall be able to use all the functions from the DLL.

### **/LF20/C++ usability**

The wrapper shall be able to be used in C++.

### **/LF30/JavaScript Functions**

The JavaScript wrapper shall be able to manipulate AML files using the functions contained inside the DLL.

### **/LF40/JavaScript usability**

The wrapper shall be able to be used in JavaScript.

### **/LF50/Import**

The console application shall be able to import and read AML files.

### **/LF60/(De-)Compression**

The console application shall be able to compress files into AMLX files and decompress them.

### **/LF70/Validation**

The console application shall be able to detect errors in the AML files and report errors with corresponding line numbers and error messages.

# Product Data

The software of this project will work with the following files.

## **/LD10/AML file**

The wrappers and the console application shall be able to successfully import and use AML files.

## **/LD20/AMLX file**

The console application shall be able to successfully decompress, use and compress AMLX files.

# Other Product Characteristics

This section describes the already known non-functional requirements for the product.

## **/NF10/Documentation**

The Documentation shall be designed in a way that the readers understand the usage of the wrapper and that the readers gain the ability to successfully implement the wrapper into their project.

The C++ wrapper documentation shall be able to explain the wrapper to the user.

The Documentation of the JavaScript wrapper shall be able to explain the wrapper to the user.

## **/NF20/Installation**

The console application should be able to run without any installation.

## **/NF30/Files**

The console application shall support AML files and AMLX files.

## **/NF40/System Environment**

The wrapper shall run under the Windows 10 operating system and newer windows versions. Furthermore, the wrapper and the console application need the already imported AML.Engine.dll.

# References

[1] Tools provided by AutomationML e.V.:

<https://www.automationml.org/o.red.c/tools.html> (geöffnet am 25.10.2019)

[2] Extensible Markup Language (XML) 1.0 (Fifth Edition):

<https://www.w3.org/TR/REC-xml/> (geöffnet am 25.10.2019)

[3] CAEX - IEC 62424: [https://www.plt.rwth-](https://www.plt.rwth-aachen.de/cms/PLT/Forschung/Projekte2/~ejwy/CAEX_IEC_62424/)

[aachen.de/cms/PLT/Forschung/Projekte2/~ejwy/CAEX\\_IEC\\_62424/](https://www.plt.rwth-aachen.de/cms/PLT/Forschung/Projekte2/~ejwy/CAEX_IEC_62424/) (geöffnet am 25.10.2019)